|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №11**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-20-23 | Комисарик М.А. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

СОДЕРЖАНИЕ

**РТУ МИРЭА** 1

СОДЕРЖАНИЕ 2

[1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ 3](#_Toc197517682)

[2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ 4](#_Toc197517683)

[ЗАКЛЮЧЕНИЕ 5](#_Toc197517684)

# ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## WebView

WebView в Android — это компонент, который позволяет разработчикам показывать веб-страницы внутри приложения. Это своего рода встроенный браузер, управляемый при помощи класса WebView. Он используется для отображения веб-контента, такого как пользовательские страницы для авторизации, справочные материалы или даже некоторые виды интерактивного контента внутри приложения.

Основные функции и использование WebView:

* отображение веб-страниц: можно загружать HTML-страницы из интернета или загружать их как локальные ресурсы;
* интерактивность: пользователи могут взаимодействовать со страницами, как в обычном веб-браузере;
* JavaScript: WebView поддерживает выполнение JavaScript, что позволяет взаимодействовать с веб-страницей и обрабатывать пользовательский ввод;
* настройка и безопасность: разработчики могут настраивать поведение WebView, включая управление кэшированием, cookies, историей просмотров и т.д. Важно правильно настроить параметры безопасности, чтобы избежать уязвимостей.

Рассмотрим пример использования WebView.

Сначала идёт добавление WebView в XML макет Activity (Рисунок 1).

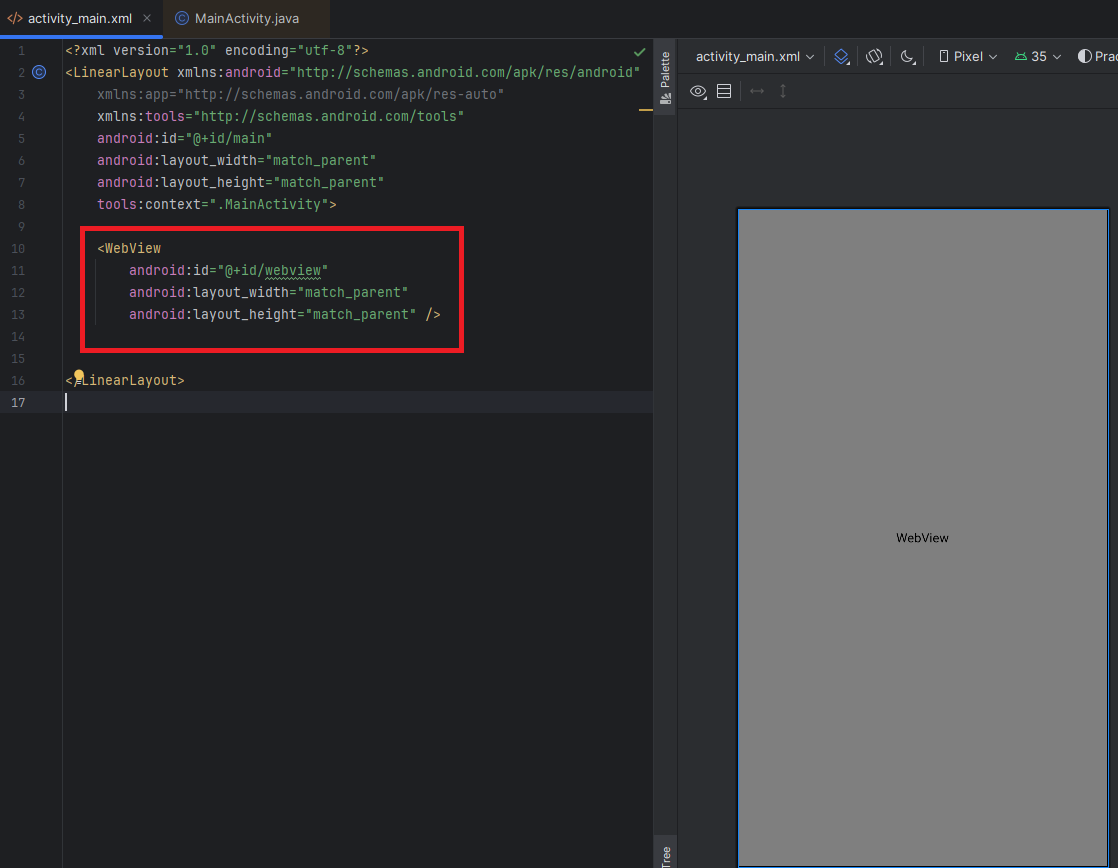


Рисунок 1 – Добавление WebWiev в XML макет Activity

Далее необходимо настроить WebView в коде Activity (Рисунок 2).

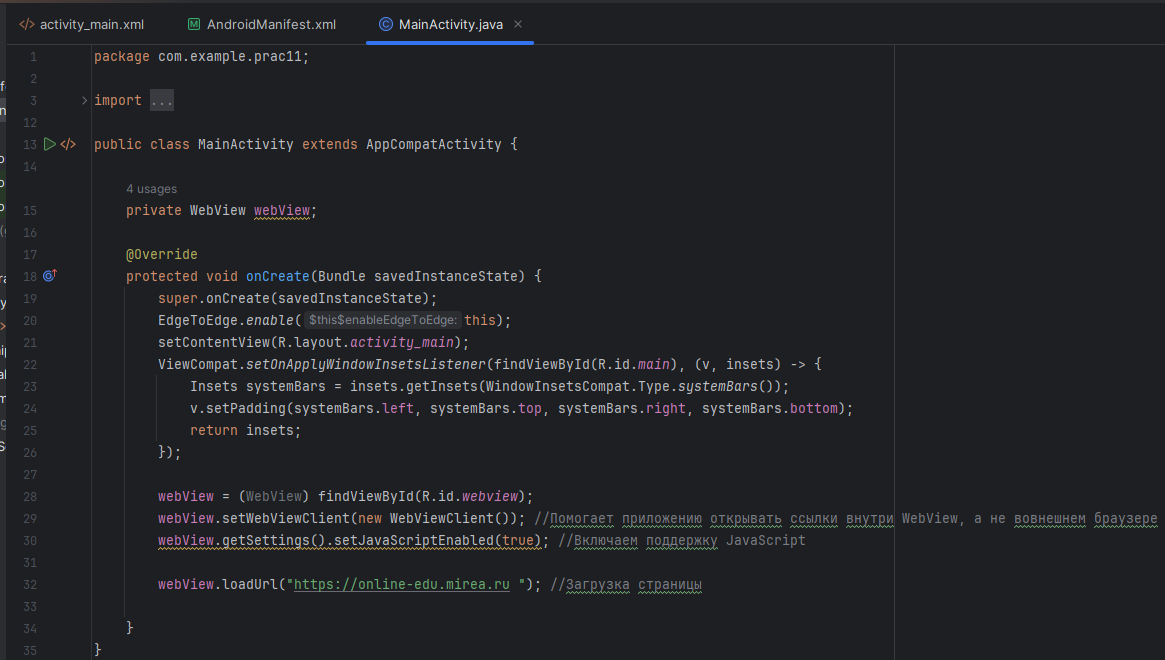


Рисунок 2 – Настройка WebView в коде Activity

Важные моменты:

* WebViewClient: этот класс помогает управлять загрузкой различных URL и следить за процессом загрузки веб-страницы;
* JavaScript: включение JavaScript необходимо для многих современных веб-страниц, которые зависят от клиентских скриптов для пользовательской интерактивности.

Для получения доступа к интернету из приложения, необходимо указать в файле манифеста AndroidManifest.xml соответствующее разрешение (Рисунок 3).

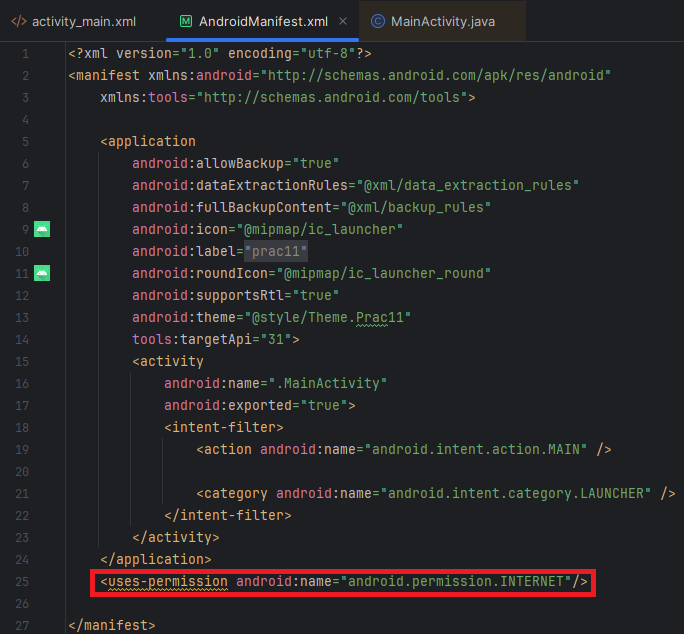


Рисунок 3 – Добавление разрешения на выход в интернет

При запуске приложения отобразится выбранный сайт (Рисунок 4).

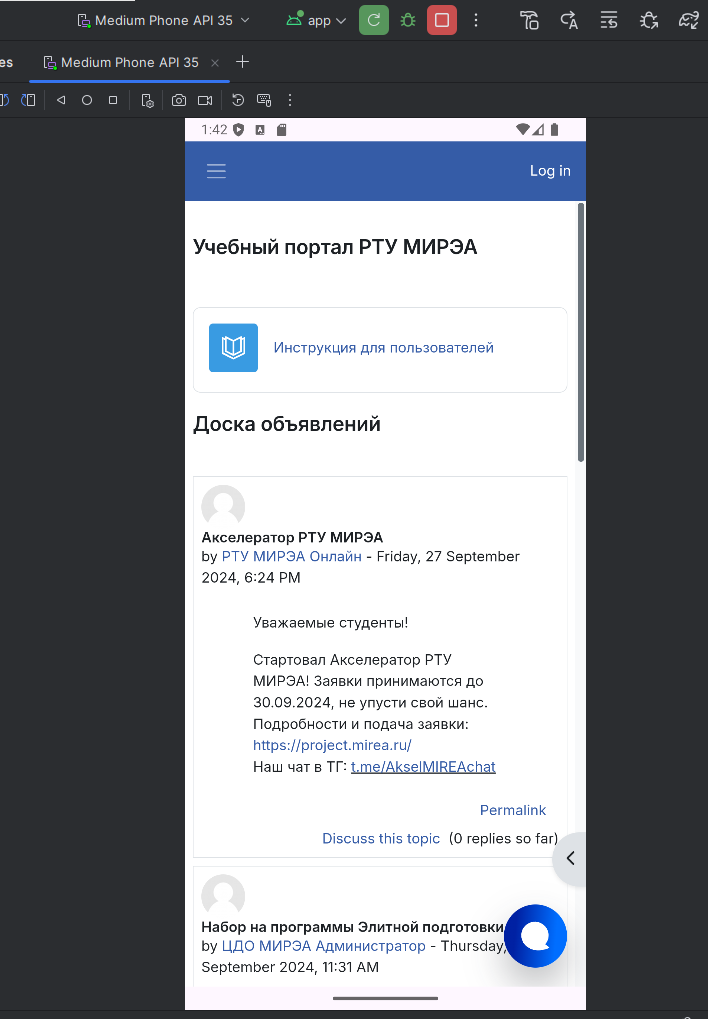


Рисунок 4 – Отображение выбранного сайта в WebView

## Работа с мультимедиа

Работа с мультимедиа в Android — это важная часть разработки мобильных приложений, так как позволяет включать в приложения такие функции, как воспроизведение музыки, видео, захват изображений и видео с камеры, и обработка звука. Это улучшает интерактивность и функциональность приложения. Android предоставляет класс MediaPlayer для воспроизведения медиафайлов. Эти файлы могут быть локальными (хранятся на устройстве) или удаленными (доступны через интернет). Ранее был рассмотрен пример воспроизведения локального файла, теперь рассмотрим воспроизведение удаленного файла.

В код activity необходимо добавить логику создания и управления MediaPlayer (Рисунок 5).

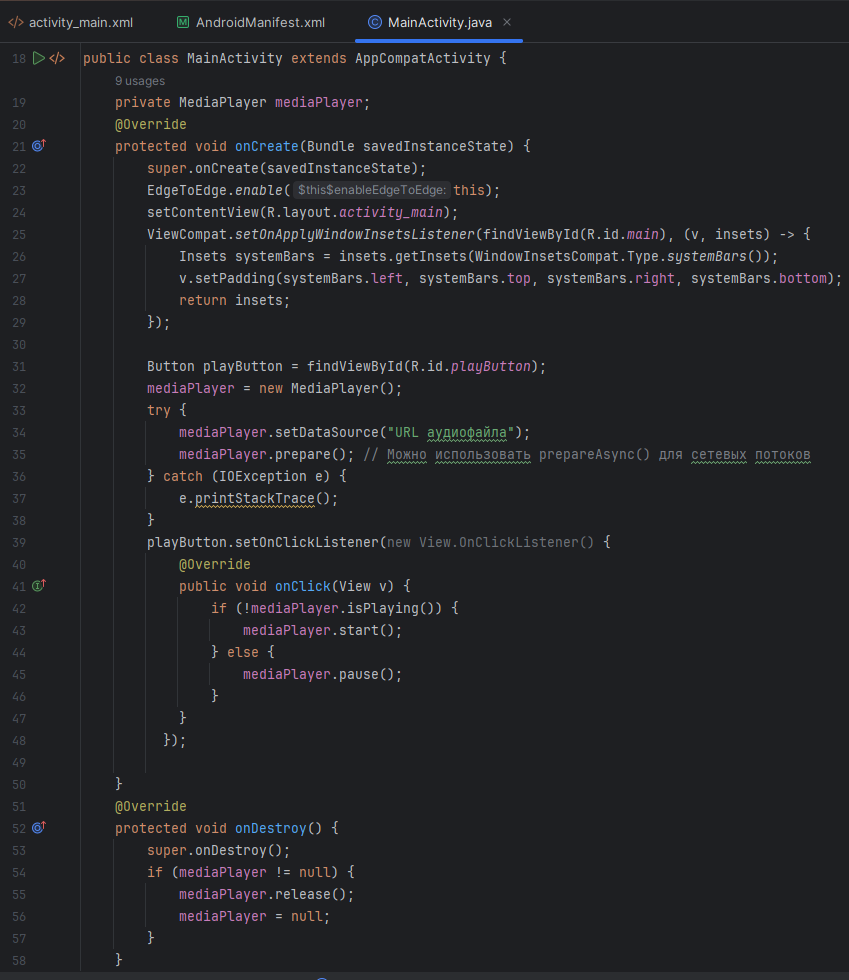


Рисунок 5 – Логика работы MediaPlayer

Этот пример демонстрирует базовый способ воспроизведения аудио с использованием MediaPlayer. Следует обратить внимание на обработку исключений и корректное освобождение ресурсов в методе onDestroy().

Работая с мультимедиа, важно помнить о потенциальных исключениях и о необходимости управления ресурсами, чтобы избежать утечек памяти и сохранить стабильность приложения.

## Анимации в Android

Анимации в Android используются для улучшения пользовательского интерфейса путём добавления визуальных эффектов при взаимодействии с элементами приложения. Они могут помочь сделать приложение более динамичным и интересным, а также интуитивно понятным, подчеркивая изменения состояний, переходы между активностями или фрагментами и реакции на действия пользователя.

Типы анимации в Android:

* View анимации: это простые анимации, которые можно применять к элементам интерфейса (views), такие как повороты, масштабирование, смещения и прозрачность. Они описываются в XML и могут быть запущены в коде;
* анимация свойств (Property Animations): эти анимации предоставляют более сложный контроль, позволяя анимировать любое свойство объекта, такое как цвет фона, позиция или размер. Наиболее популярным классом для таких анимаций является ObjectAnimator;
* анимации переходов: используются для анимации переходов между активностями или фрагментами, позволяя создавать сложные анимации при переключении между экранами.

Рассмотрим несколько примеров анимации элементов.

### Анимация вращения элемента

Для анимации вращения элемента необходимо воспользоваться таким классом, как ObjectAnimator (Рисунок 6).

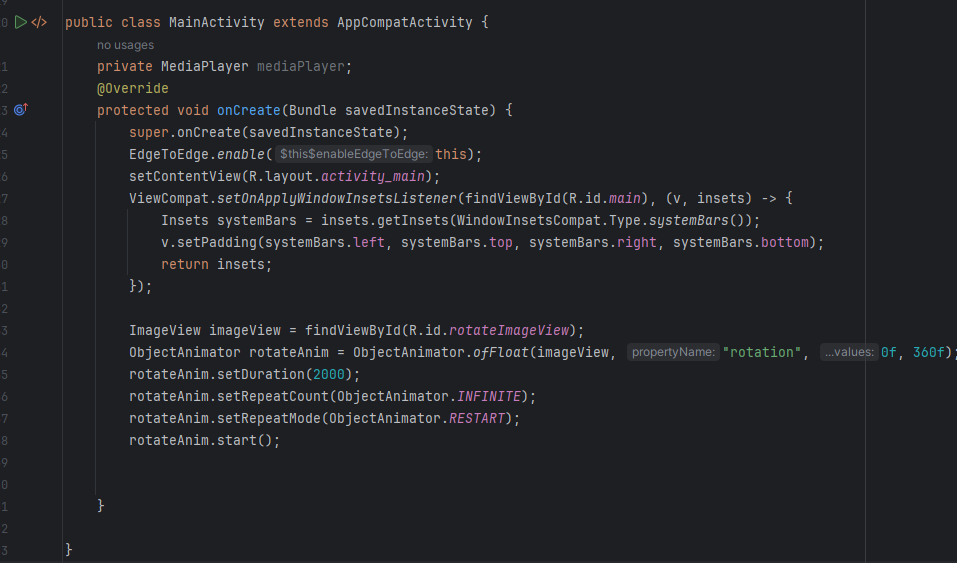


Рисунок 6 – Логика анимации вращения элемента

ObjectAnimator – это подкласс ValueAnimator, который позволяет нам устанавливать целевой объект и свойство объекта для анимации. Это простой способ изменить свойства вида с указанной продолжительностью. Мы можем указать конечную позицию и продолжительность анимации.

Методы:

* ofFloat(): создает и возвращает ObjectAnimation, который анимирует координаты;
* setDuration(): устанавливает продолжительность анимации;
* getRepeatCount() – определяет, сколько раз должна повторяться анимация;
* getRepeatMode() – определяет, что должна делать эта анимация, когда она дойдет до конца.

При запуске приложения выбранный ImageView начнёт вращаться по часовой стрелке (Рисунки Рисунок 7-Рисунок 8).

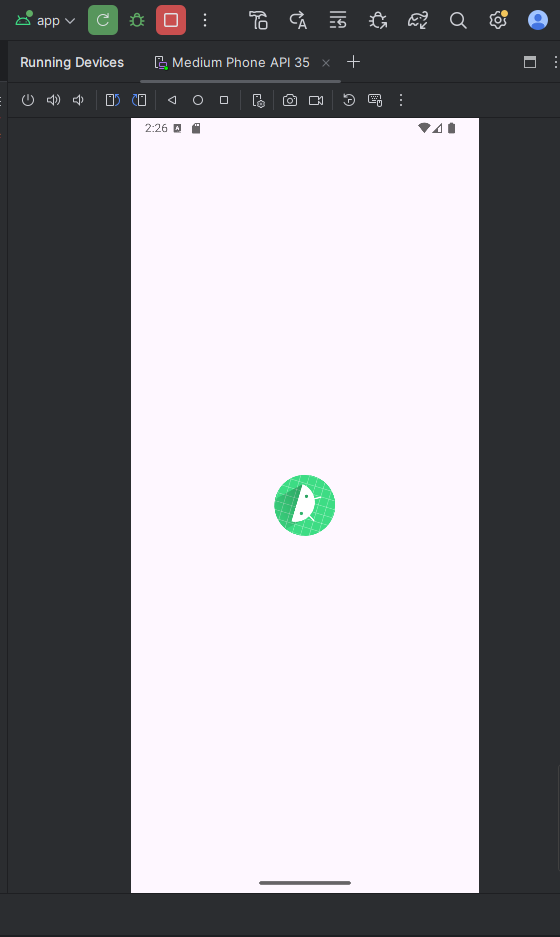


Рисунок 7 – Отображение вращения ImageView, часть 1

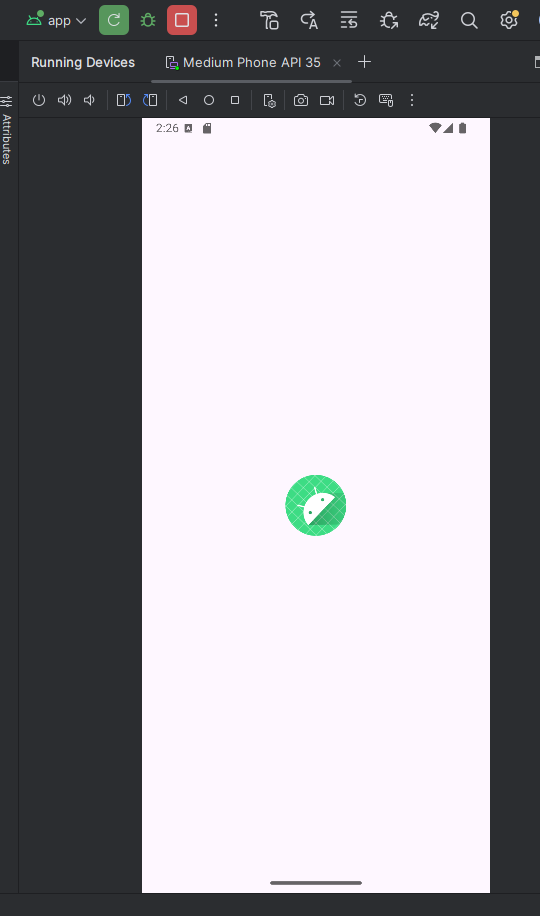


Рисунок 8 – Отображение вращения ImageView, часть 2

### Анимация перемещения элемента по экрану

В данном случае также будет использоваться ObjectAnimator с методами, описанными в пункте 1.3.1 (Рисунок 9).

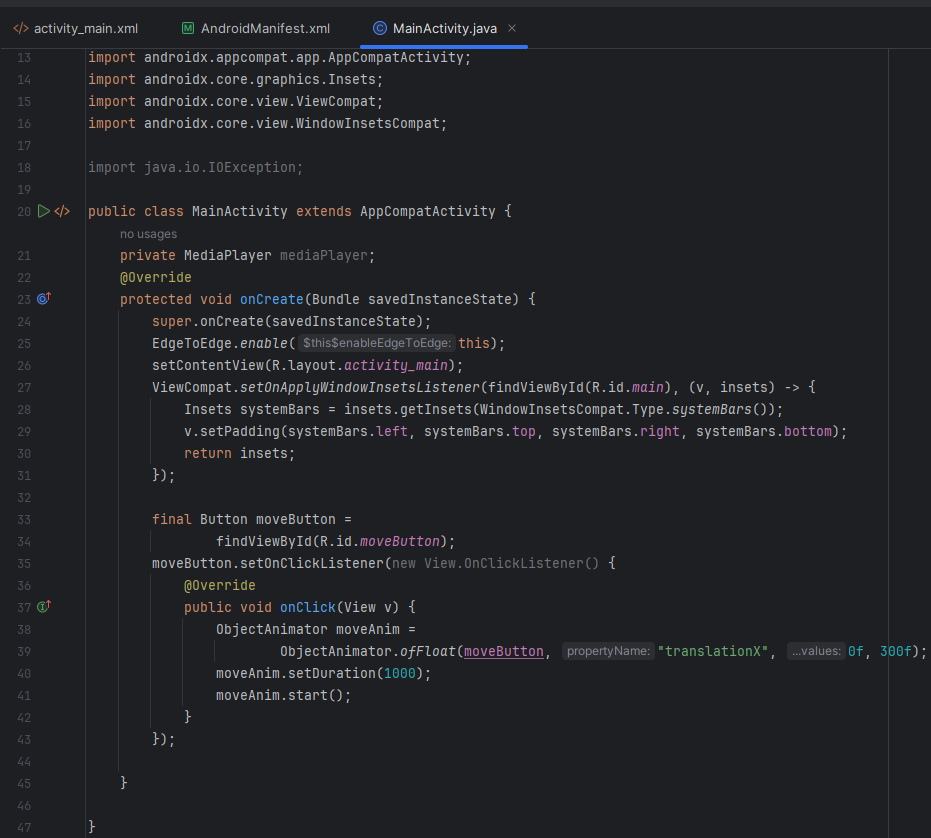


Рисунок 9 – Логика анимации перемещения элемента по экрану

При нажатии на кнопку она переместится вправо на заранее прописанное расстояние (Рисунки Рисунок 10-Рисунок 11).

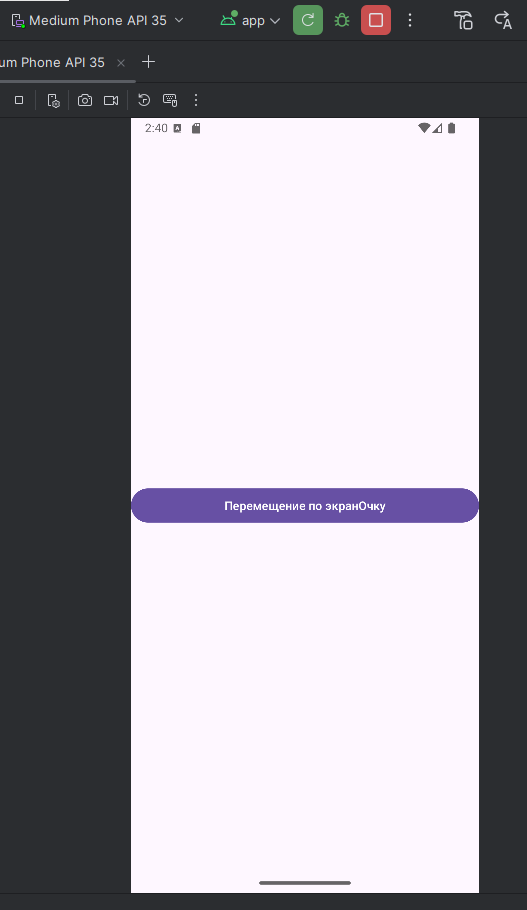


Рисунок 10 – Отображение анимации перемещения элемента, часть 1

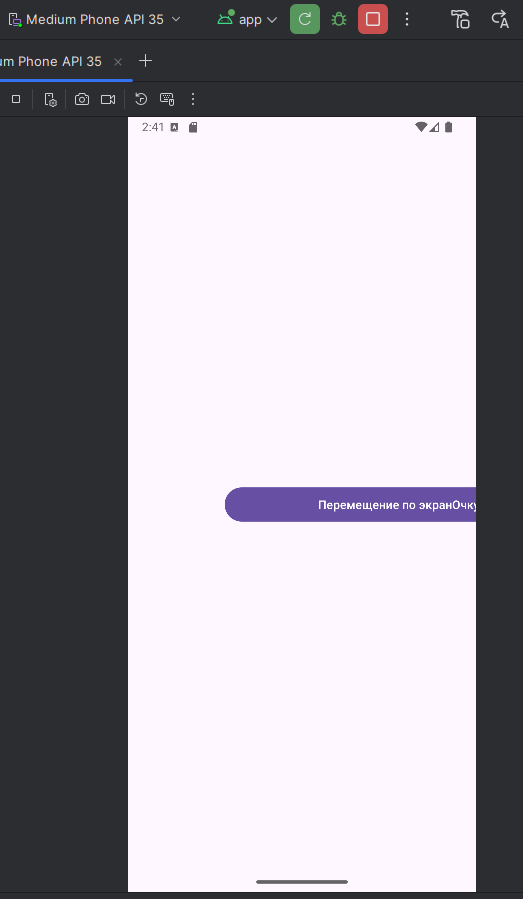


Рисунок 11 – Отображения анимации перемещения элемента, часть 2

### Анимация изменения размера элемента

И последняя на рассмотрение анимация – анимация изменения размера элемента (Рисунки Рисунок 12-Рисунок 14).

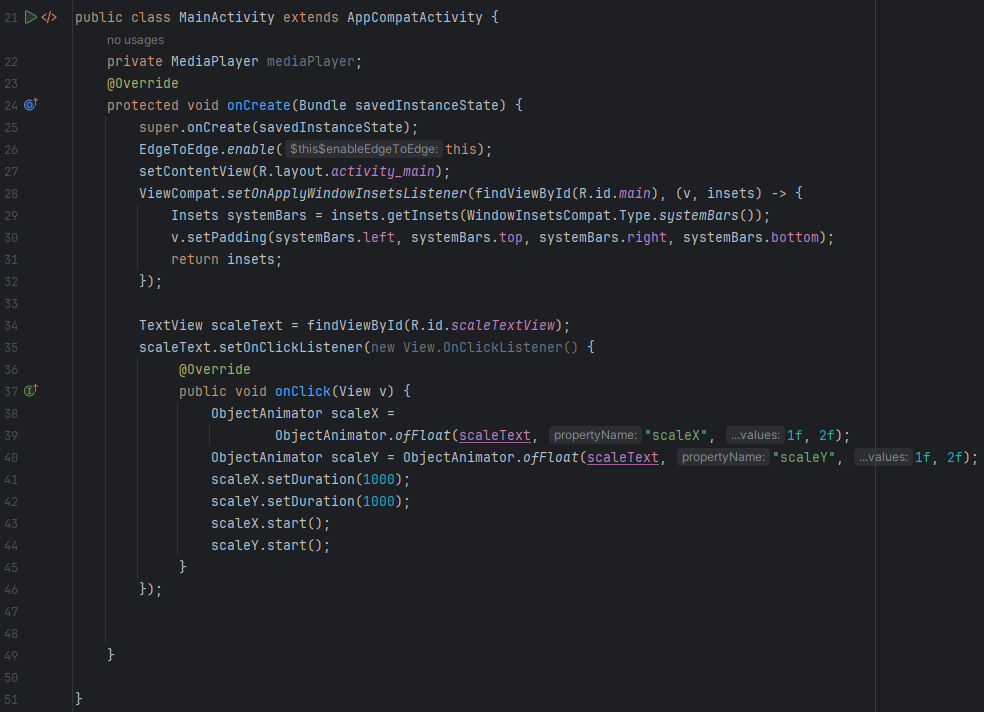


Рисунок 12 – Логика анимации изменения размера элемента

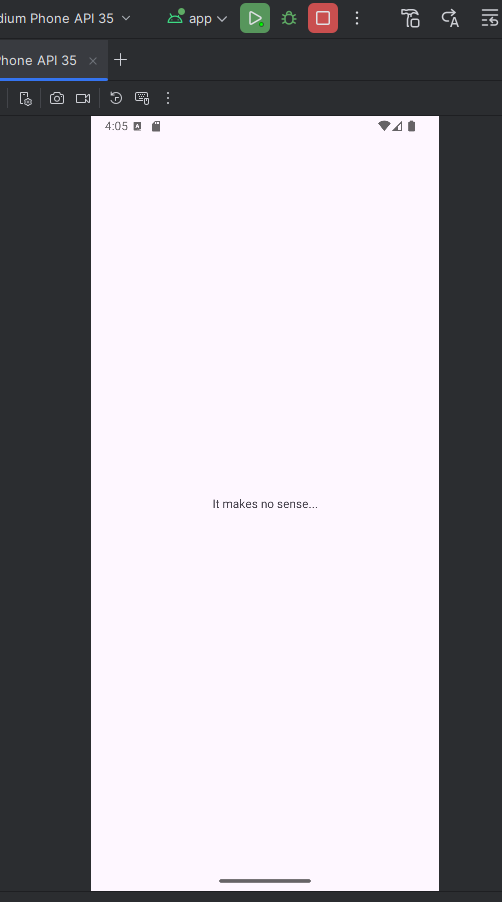


Рисунок 13 – Отображение анимации изменения размера элемента, часть 1

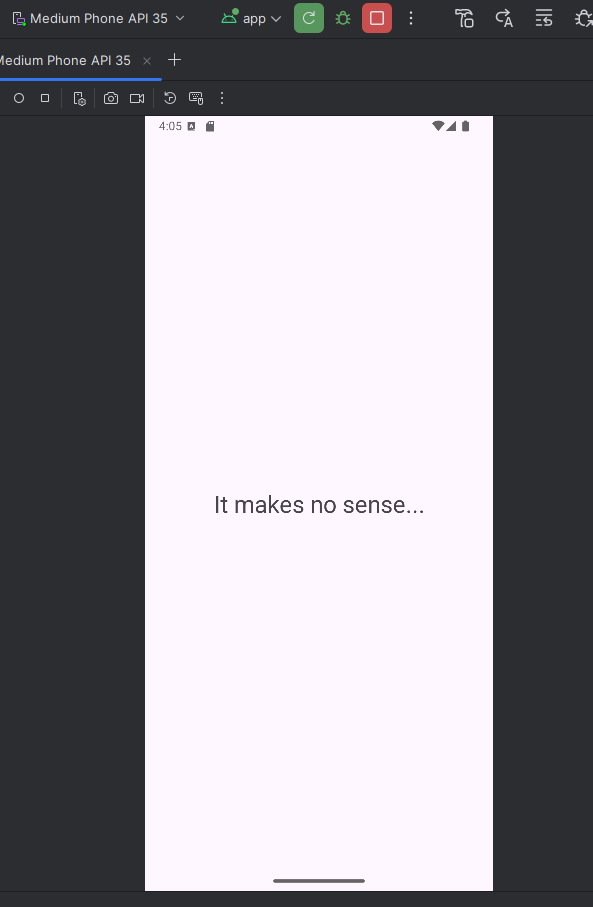


Рисунок 14 – Отображение анимации изменения размера элемента, часть 2

Эти примеры демонстрируют различные способы использования анимации для улучшения визуального восприятия и интерактивности Androidприложений.

Анимацию можно создавать как отдельный ресурс в папке под названием «anim» для хранения ресурсов анимации.

## Уведомления

Уведомления в Android представляют собой сообщения, которые можно отображать вне интерфейса вашего приложения для информирования пользователей о различных событиях, даже когда приложение не используется активно. Это мощный инструмент для повышения взаимодействия и поддержания актуальности приложения в глазах пользователей.

Уведомления в Android управляются через NotificationManager, который является системной службой, ответственной за доставку уведомлений. Для создания уведомления используется Notification.Builder класс, который позволяет настроить заголовок, текст, иконку и другие параметры уведомления. С Android Oreo (API уровень 26) введено понятие каналов уведомлений (notification channels), которое требует определения категории уведомлений, что позволяет пользователям управлять настройками уведомлений для различных типов контента.

Для отправки уведомлений на Android версии 13+ необходимо в файле манифеста прописать соответствующее разрешение (Рисунок 15).



Рисунок 15 – Разрешение приложения на отправку уведомлений в манифесте

Далее создадим код для показа простых уведомлений (Рисунок 16).

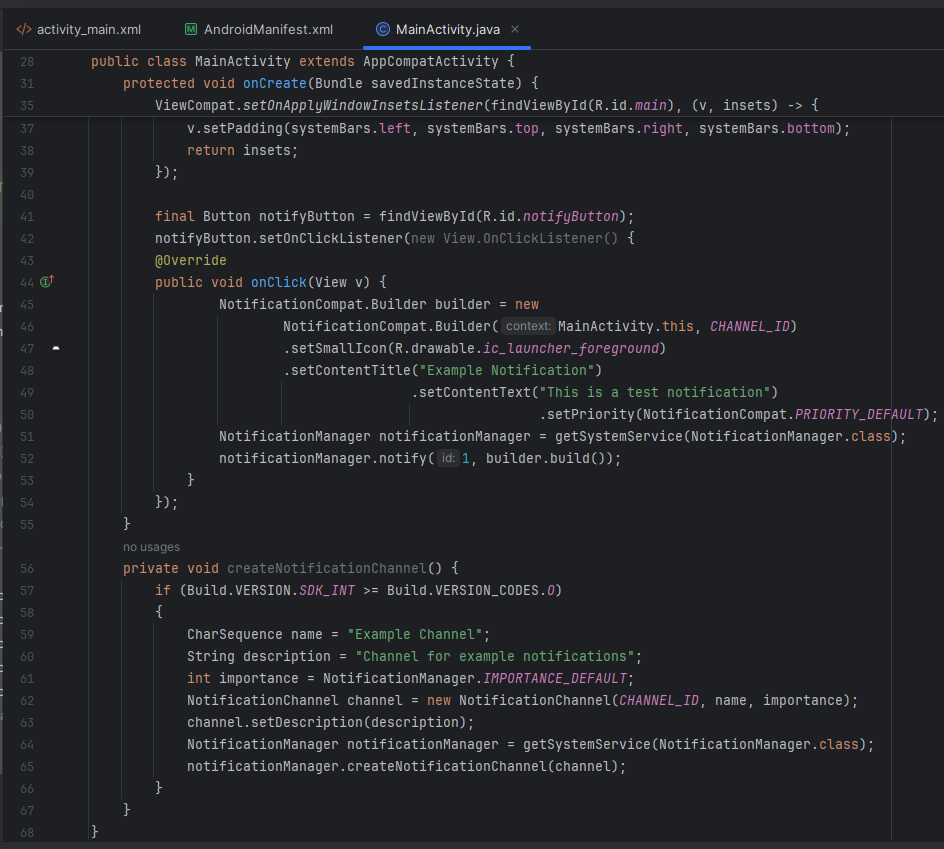


Рисунок 16 – Логика отправки уведомлений пользователю

В этом примере создается канал уведомлений, что необходимо для API 26 и выше. Затем, при нажатии на кнопку, создается уведомление с использованием NotificationCompat.Builder, которое отображает текст и иконку (Рисунок 17). Уведомление отправляется через NotificationManager.

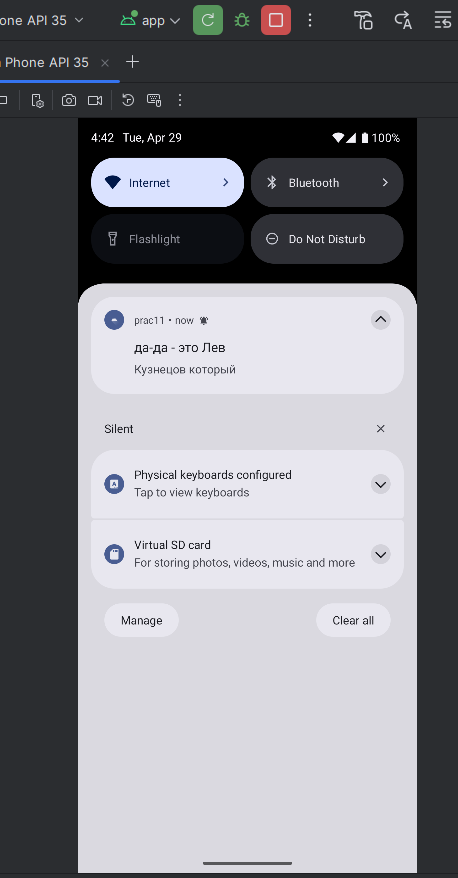


Рисунок 17 – Отображение отправленного уведомления

Используемая переменная CHANNEL\_ID должна быть задана глобально в классе private static final String CHANNEL\_ID = "example\_channel".

Отложенные уведомления в Android позволяют отправить уведомление через определённый период времени, что может быть полезно для напоминаний, задач по расписанию или для других сценариев, где требуется не немедленное, а запланированное уведомление. Для реализации отложенных уведомлений можно использовать AlarmManager, который позволяет запланировать выполнение кода в определённое время (Рисунки Рисунок 18-Рисунок 19).

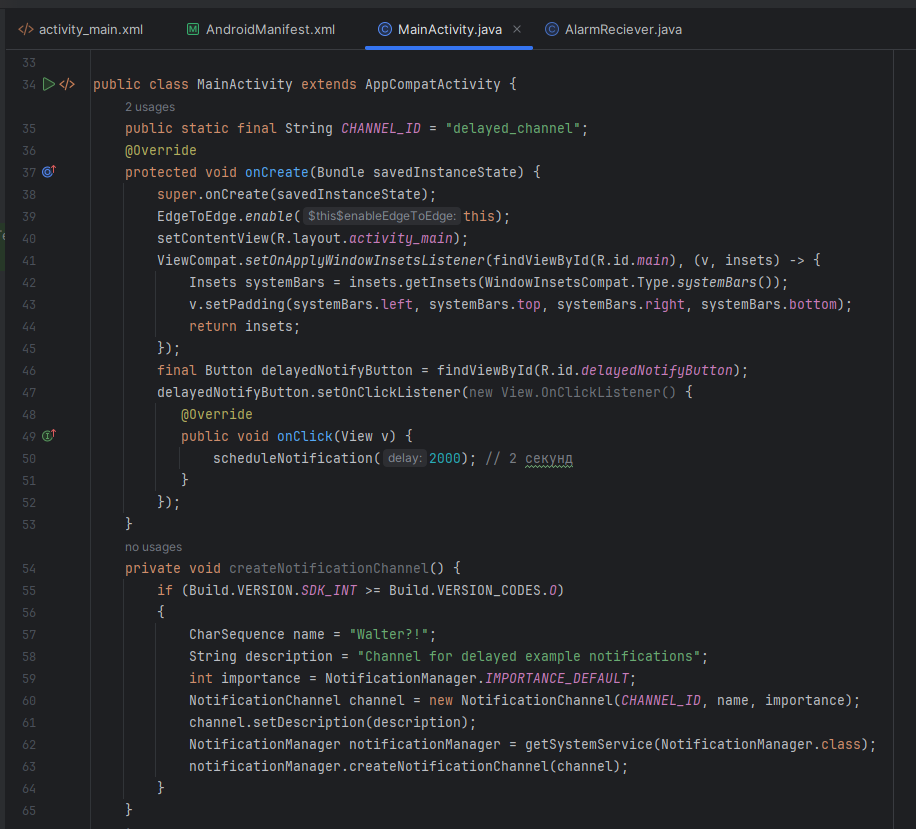


Рисунок 18 – Описание логики отправки сообщений с задержкой, часть 1

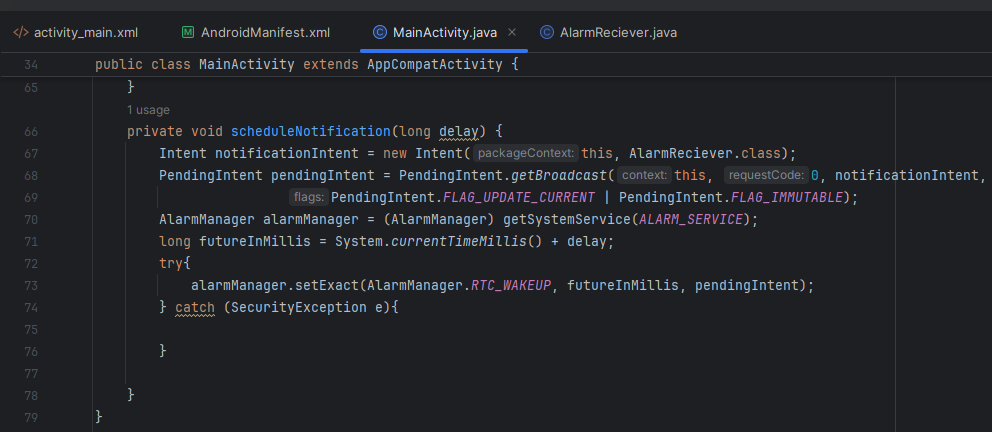


Рисунок 19 – Описание логики отправки сообщений с задержкой, часть 2

Также для корректной работы приложения необходимо описать вспомогательный класс – AlarmReciever (Рисунок 20).

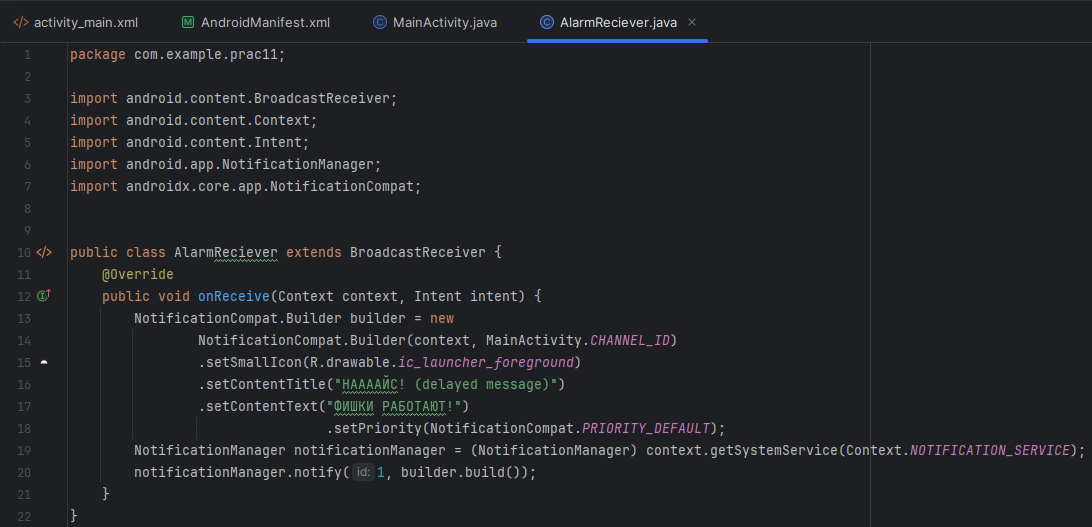


Рисунок 20 – Описание класса AlarmReceiever

При нажатии на кнопку на экране через 2 секунды появляется оповещение (Рисунок 21).

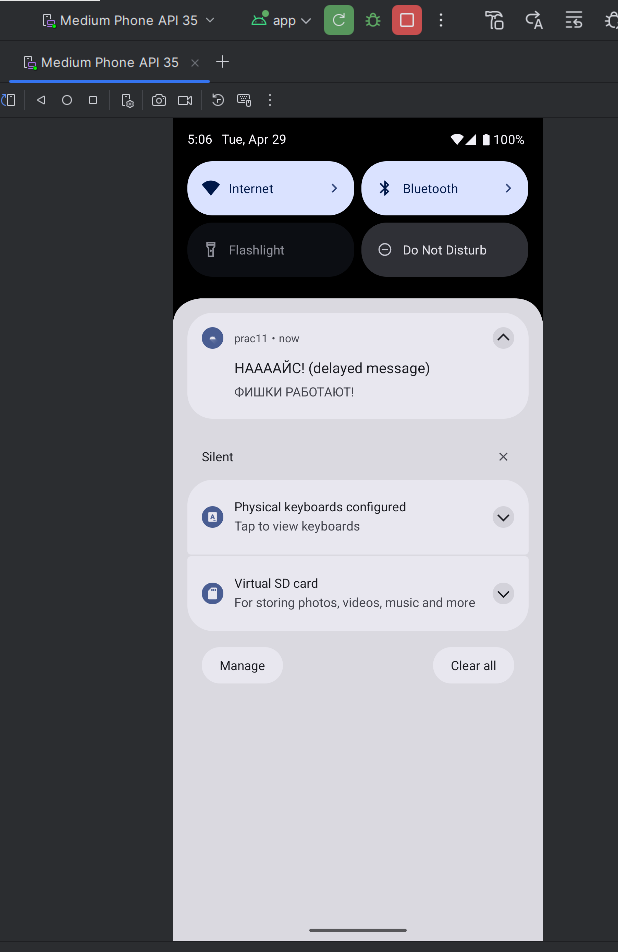


Рисунок 21 – Отображение оповещения с задержкой

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ

ЗАКЛЮЧЕНИЕ

В ходе работы были получены знания по работе с x, а также x в Android Studio. Полученные знания были закреплены путём x, а также x.